java.util.function

Java provides a library of functional interfaces in the java.util.function package.

We looked at one already, the Consumer interface.

I'll look at another of these interfaces now, the BinaryOperator, in code.

The Four basic categories of Functional Interfaces in java.util.function package

It's a good idea to know the four basic types of functional interfaces in the java.util.function package.

There are over forty interfaces in this package.

These can all be categorized as one of the following types.

This slide shows the four categories, with the simplest method shown.

Interface Category	Basic Method Signature	Purpose
Consumer	`void accept(T t)`	execute code without returning data
Function	`R apply(T t)`	return a result of an operation or function
Predicate	`boolean test(T t)`	test if a condition is true or false
Supplier	`T get()`	return an instance of something

The Consumer interface

On this slide, I'm showing the two most common Consumer interfaces, and the functional method on each.

The Consumer interface takes one argument of any type.

The BiConsumer interface takes two arguments, of two different types.

Interface Name	Method Signature
Consumer	`void accept(T t)`
BiConsumer	`void accept(T t, U u)`

A Consumer Lambda Expression Example

This slide shows an example consumer lambda expression. It takes one argument and executes a single statement.

No result is returned.

Example Lambda Expression for Consumer	Consumer Method
`s -> System.out.println(s)`	`void accept(T t)`

The Predicate Interface

The predicate interfaces take one or two arguments, and always returns a boolean value.

They are used to test a condition, and if the condition is true, some operation will be performed.

Interface Name	Method Signature
Predicate	`boolean test(T t)`
BiPredicate	`boolean test(T t, U u)`

A Predicate Lambda Expression Example

In this example, the expression takes a String, and tests if it's equal to the literal text, “Hello” here, ignoring case, so it returns either true or false.

Example Lambda Expression for Consumer
`s -> s.equalsIgnoreCase("Hello")`